

# The Mechanization of Mathematics

BAMA

San José

October 19, 2005

Copyright, 2005 Michael Beeson

# John Henry:

*I'll die 'fore I'll let that steam  
drill beat me down!*





# Kasparov vs. Deep Blue (1997):

- ◆ In a dazzling hour-long game , the Deep Blue IBM computer demolished an obviously overwhelmed Garry Kasparov and won the six-game man-vs.-machine chess match.



# The New York Times

ON THE WEB

- **Computer Math Proof Shows Reasoning Power**
  - **By GINA KOLATA**
  - **December 10, 1996**
- ◆ Computers are whizzes when it comes to the grunt work of mathematics. But for creative and elegant solutions to hard mathematical problems, nothing has been able to beat the human mind. That is, perhaps, until now.



# The New York Times

ON THE WEB

- ◆ A computer program written by researchers at Argonne National Laboratory in Illinois has come up with a major mathematical proof that would have been called creative if a human had thought of it. In doing so, the computer has, for the first time, got a toehold into pure mathematics, a field described by its practitioners as more of an art form than a science.



- ◆ Dr. William McCune at Argonne Labs, Illinois, in his office. The “Robbins Conjecture” proof is on the screen.
- ◆ **Photo credit: Lloyd DeGrane / The New York Times**

# The Robbins Conjecture

- ◆ Prove that algebras satisfying these axioms are Boolean:
  - ◆  $x + y = y + x$ .
  - ◆  $(x + y) + z = x + (y + z)$ .
  - ◆  $n(n(x + y) + n(x + n(y))) = x$ .
- ◆ It's enough to prove
- ◆  $n(n(x) + y) + n(n(x) + n(y)) = x$ .  
[Huntington equation]
- ◆ Eight days, 30 megabytes



# Pre-Computer History

- ◆ Leibniz
- ◆ Boole
- ◆ Frege
- ◆ Russell
- ◆ Hilbert
- ◆ Turing

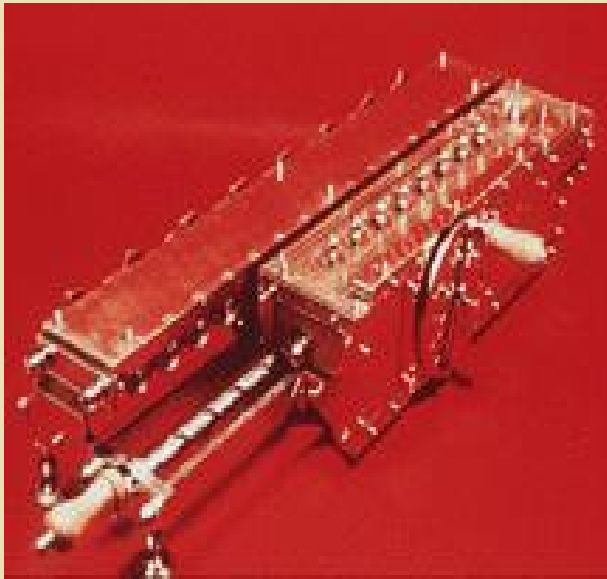


# Leibniz (1646-1716)

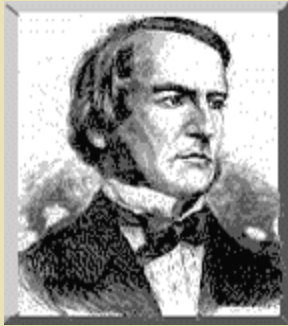


- ◆ *Calculemus*
- ◆ “Let us calculate”
- ◆ Envisioned a formal language to reduce reasoning to calculation.

# Leibniz's Stepped Reckoner



- ◆ The first calculating machine that could add, subtract, divide, and multiply.
- ◆ Pascal's could only add and subtract.
- ◆ Now in museums in Munich and Hanover.



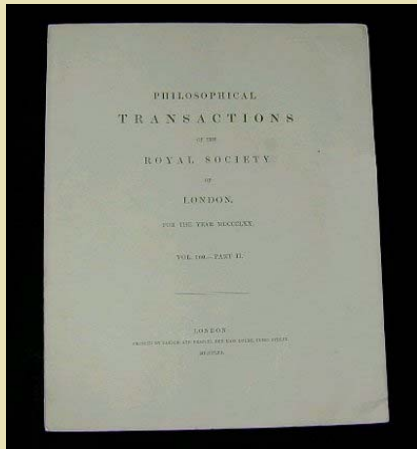
# George Boole (1815-1864)

- ◆ Wrote a book:  
*Laws of Thought*
- ◆ The laws are now called Boolean Algebra

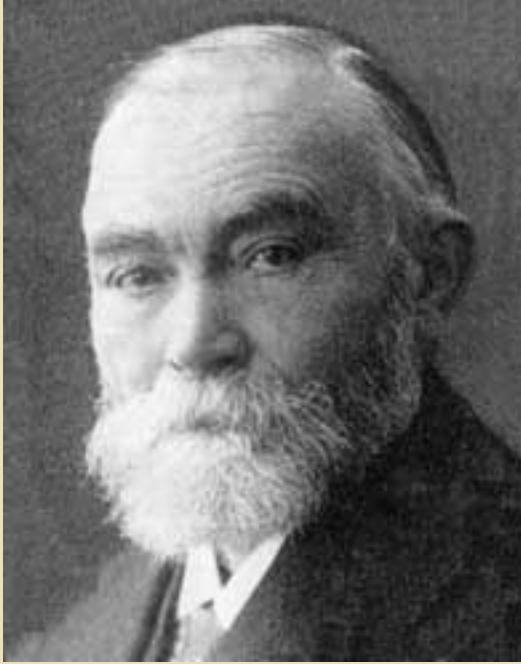


# William Stanley Jevons (1835-1882)

- ◆ Built the Logical Piano
- ◆ First machine to do mechanical inference
- ◆ Museum of Science at Oxford
- ◆ Implemented Boole's rules
- ◆ *On the Mechanical Performance of Logical Inference* read before Royal Society (1870)

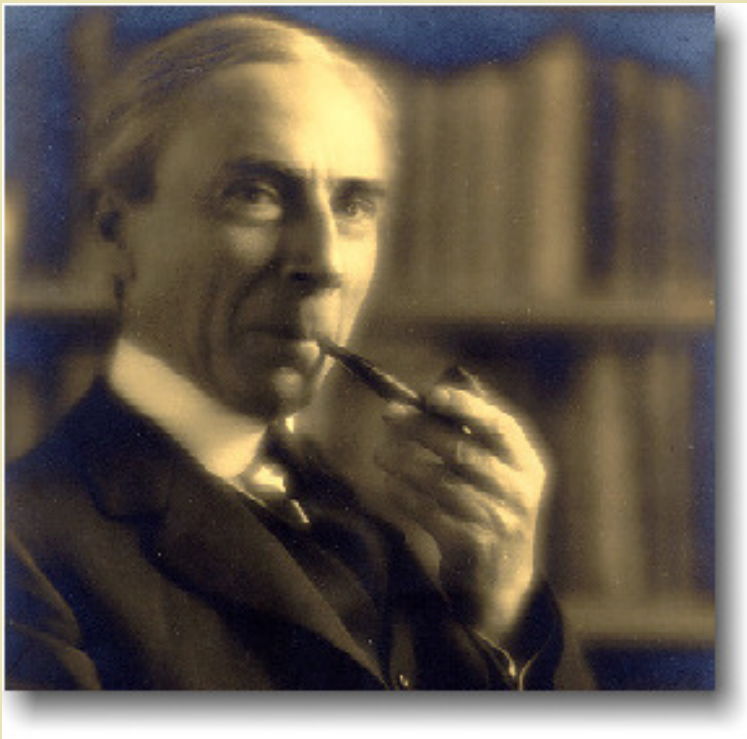






# Gottlob Frege (1848-1925)

- ◆ Created modern logic including “for all”, “there exists”, and rules of proof.
- ◆ Tried to reduce mathematics to logic, including the concept of number.
- ◆ *Begriffsschrift* published in 1879, when Frege was 31 years old.
- ◆ “a symbolic language of pure thought, modeled upon that of arithmetic.”



# Bertrand Russell (1872-1970)

- ◆ Found Frege's error
- ◆ Russell paradox:  $R = \{ x : x \notin x \}$ . The existence of  $R$  leads to a contradiction, but Frege said  $\{ x : P(x) \}$  exists for any  $P$ .
- ◆ Wrote *Principia Mathematica* to save mathematics from this contradiction.



# David Hilbert (1862-1943)

- ◆ Development of formal logic
- ◆ Reductionist program: Express mathematics in logic, give decision methods for logic
- ◆ Posed the *Entscheidungsproblem* (decision problem) for logic.
- ◆ 1928, in Hilbert-Ackermann



# Entscheidungsproblem

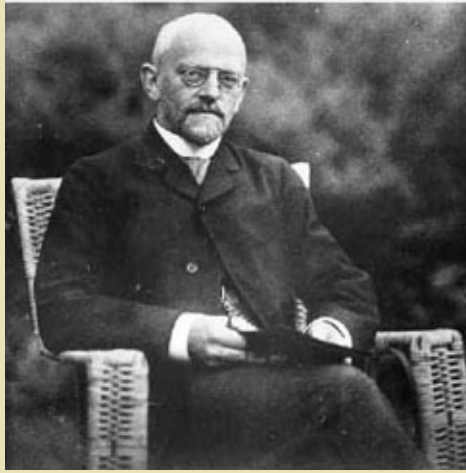
- ◆ Is there an “decision algorithm” such that
- ◆ It takes two inputs: a set of axioms, and a conjecture.
- ◆ It computes for a finite time and outputs either a proof of the conjecture from the axioms, or “no proof exists”.
- ◆ The result is always correct.
- ◆ ?





# Getting the problem stated right

- ◆ What are axioms?
- ◆ What is a proof?
- ◆ What is an algorithm?



# Tables, chairs, and beer mugs

- ◆ Hilbert contributed to answering the questions about axioms and proofs.
- ◆ Famous book *Foundations of Geometry* (1899) provided a careful axiomatic reworking of Euclid from 21 axioms.
- ◆ If you replace “points, lines, and planes” by “tables, chairs, and beer mugs”, the reasoning should still be correct.



# Opposition from Poincaré (1908)


Thus it will be readily understood that in order to demonstrate a theorem, it is not necessary or even useful to know what it means. We might replace geometry by the reasoning piano imagined by Stanley Jevons, or...a machine where we should put in axioms at one end and take out theorems at the other, like that legendary machine in Chicago where pigs go in alive and come out transformed into hams and sausages.



## Alan Turing (1912-1954)

- ◆ Answered “What is an algorithm” by defining Turing machines.
- ◆ Solved the “halting problem”--there is no Turing machine that takes as inputs a Turing machine  $M$  and an input  $x$  for  $M$ , and determines correctly whether  $M$  halts on input  $x$ .





# Turing's solution of the Entscheidungsproblem

- ◆ Write down axioms  $A$  to describe the computations of Turing machines.
- ◆ Machine  $M$  halts at input  $x$  iff  $A$  proves the theorem “ $M$  halts at input  $x$ ”.
- ◆ If we had an algorithm to determine the consequences of axioms  $A$ , it would solve the halting problem.

# Alonzo Church (1903-1995)



- ◆ Invented  $\lambda$ -calculus to answer “what is an algorithm?”
- ◆ Also solved the Entscheidungsproblem, using  $\lambda$ -calculus (1936).
- ◆ Arithmetic is undecidable.



# “Negative Results” of 1930’s

- ◆ Halting problem
- ◆ Entscheidungsproblem (1936)
- ◆ Gödel’s incompleteness theorem
- ◆ *Mathematics cannot be completely mechanized*



# Possible Loopholes

- ◆ Decision procedure could work for a *particular* axiom system to say if any formula is a theorem.
- ◆ Algorithm could *sometimes* tell us whether a formula is a theorem or not (but not *always*).
- ◆ Algorithm could work sometimes for a particular axiom system.



# The first theorem-provers

- ◆ Davis (1954) Presburger arithmetic. Proved the sum of two even numbers is even.
- ◆ Newell, Shaw, and Simon (1957) *Logic Theorist*. Proved propositional logic theorems in the system of *Principia*.
- ◆ Gelernter's geometry prover (1959) used a “diagram” to prune false goals.





## The 1960's

- ◆ Davis-Putnam procedure. Skolem functions and conjunctive normal form.
- ◆ Wang (1963) Program proved all 400 pure predicate-calculus theorems in *Principia*.
- ◆ *Photo 1982, showing Martin Davis, Julia Robinson, Yuri Matiyesevich*





# Resolution

- ◆ J. A. Robinson (1963)
- ◆  $(p \mid q)$  means “p or q”
- ◆  $\neg p$  means “not p”
- ◆ Resolution rule:  
From  $(p \mid q)$  and  $(\neg p \mid r)$  deduce  $(q \mid r)$

# Unification

- ◆ Published by J. A. Robinson (1965)
- ◆ “in the air”--implemented by others as early as 1962.
- ◆ Purpose of unification algorithm: find values of variables to make two terms match.
- ◆ Example: given  $f(x,g(x))$  and  $f(g(c),z)$  we find  $x = g(c)$ ,  $z = g(g(c))$



# Resolution and Unification

- ◆ From  $(p \mid q)$  and  $(-s \mid r)$  infer  $(q^* \mid r^*)$  provided  $p^* = -s^*$
- ◆ Here  $*$  means the substitution resulting from unifying  $p$  and  $-s$  successfully.



# Searching for proofs

- ◆ Start with the axioms and the negated goal.
- ◆ Perform resolutions (using unification) until a contradiction is reached.
- ◆ Millions of clauses may be generated. Often memory or time will run out before you get a proof.

# Kinds of Mathematical Reasoning

- ◆ Purely logical
- ◆ Equational, as in the Robbins problem, or in group or ring theory.
- ◆ Single-theory, as in geometry
- ◆ Uses calculations, as in algebra or calculus
- ◆ Uses mathematical induction
- ◆ Uses definitions (perhaps lots of them)
- ◆ Uses a little number theory and simple set theory (as in undergraduate algebra courses)



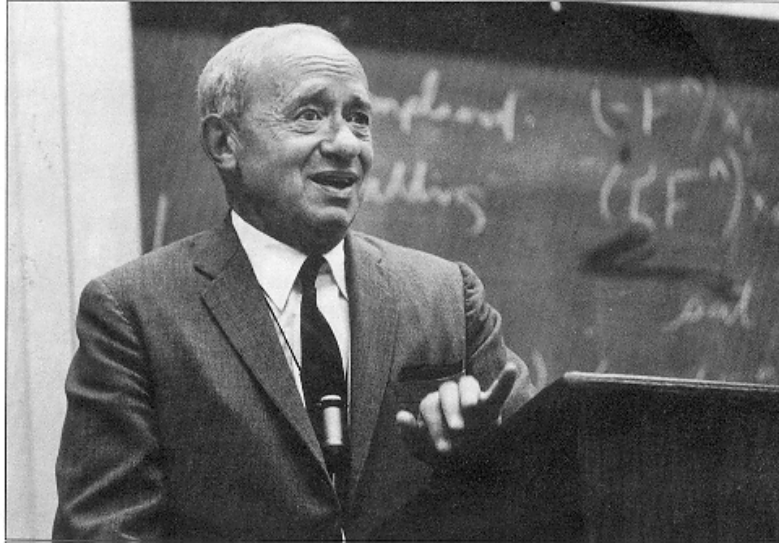
# Decision Methods

- ◆ Some branches of mathematics *can* be mechanized!
- ◆ However, often the algorithm is exponential or worse.
- ◆ There are important examples anyway.



# Alfred Tarski (1902-1983)

Photo 1971



- ◆ Real-closed fields
- ◆ questions involving  $+$ ,  $*$ ,  $<$ , and quantifiers
- ◆ Example: for which values of  $b$  does a polynomial  $f(x,b)$  have a root between 0 and 1?



# Another Example

- ◆ When is  $x^4 + ax^3 + bx^2 + c + d$  positive definite, that is, positive for all  $x$ ?
- ◆ Answer should not involve  $x$ , but only conditions on  $a, b, c, d$ .

# The Answer

$$\begin{aligned} & 256 d^3 - 192 a c d^2 - 128 b^2 d^2 + 144 a^2 b d^2 - 27 a^4 d^2 + 144 b c^2 d - 6 a^2 c^2 d - 80 a b^2 c d + 18 a^3 b c d + 16 b^4 d - 4 a^2 b^3 d - \\ & 27 c^4 + 18 a b c^3 - 4 a^3 c^3 - 4 b^3 c^2 + a^2 b^2 c^2 \geq 0 \wedge [ [ 108 \\ & c^2 - 108 a b c + 27 a^3 c + 32 b^3 - 9 a^2 b^2 > 0 \wedge 384 d^2 - 192 a c d - \\ & 128 b^2 d + 144 a^2 b d - 27 a^4 d + 72 b c^2 - 3 a^2 c^2 - 40 a b^2 c + 9 \\ & a^3 b c + 8 b^4 - 2 a^2 b^3 \leq 0 ] \vee [ 256 d^3 - 192 a c d^2 - 128 b^2 d^2 \\ & + 144 a^2 b d^2 - 27 a^4 d^2 + 144 b c^2 d - 6 a^2 c^2 d - 80 a b^2 c d + \\ & 18 a^3 b c d + 16 b^4 d - 4 a^2 b^3 d - 27 c^4 + 18 a b c^3 - 4 a^3 c^3 - 4 \\ & b^3 c^2 + a^2 b^2 c^2 > 0 \wedge 384 d^2 - 192 a c d - 128 b^2 d + 144 a^2 b d \\ & - 27 a^4 d + 72 b c^2 - 3 a^2 c^2 - 40 a b^2 c + 9 a^3 b c + 8 b^4 - 2 a^2 \\ & b^3 \geq 0 \wedge 768 d - 192 a c - 128 b^2 + 144 a^2 b - 27 a^4 \geq 0 ] \vee [ 108 \\ & c^2 - 108 a b c + 27 a^3 c + 32 b^3 - 9 a^2 b^2 \geq 0 \wedge 256 d^3 - 192 a c \\ & d^2 - 128 b^2 d^2 + 144 a^2 b d^2 - 27 a^4 d^2 + 144 b c^2 d - 6 a^2 c^2 \\ & d - 80 a b^2 c d + 18 a^3 b c d + 16 b^4 d - 4 a^2 b^3 d - 27 c^4 + 18 a b \\ & c^3 - 4 a^3 c^3 - 4 b^3 c^2 + a^2 b^2 c^2 > 0 ] ] \end{aligned}$$



# Quantifier Elimination

- ◆ Tarski's method
- ◆  $\exists x, y (f(a, x, y) = 0 \ \& \ g(a, x, y) > 0)$  can be expressed without a quantifier in the form  $h(a) = 0 \ \& \ k(a) > 0$ .
- ◆ Here  $f, g$ , and  $h$  are polynomials.
- ◆ Related to Sturm's theorem.
- ◆ *Semi-algebraic sets* are defined this way



# Geometry

- ◆ Descartes reduced geometry to algebra using coordinates.
- ◆ Tarski's decision procedure for algebra therefore works for geometry too.
- ◆ Tarski's student Szmielew made it work for hyperbolic geometry too.



# Implementation of Quantifier Elimination

- ◆ Cylindric decomposition (George Collins)
- ◆ qepcad (implemented in Mathematica 4.02)
- ◆ Worst case is double exponential (Fischer and Rabin)
- ◆ But, it's only double exponential in the number of variables if we use cylindric decomposition.



# Double Exponential Time

- ◆ Can two unit spheres can be packed without overlapping in the cube of side  $M$ ?
- ◆  $3M^2 - 6M + 2 \geq 0$  and  $M \geq 1$
- ◆ Can three unit spheres be packed without overlapping in the cube of side  $M$ ?
- ◆ *Too hard for today's computers—9 variables is too many. It is  $10^{33}$  times harder than 6 variables!*
- ◆  $2^{(2^9)} = 2^{512} = 10^{54}$
- ◆  $2^{(2^6)} = 2^{64} = 10^{19}$
- ◆ *Ratio is  $10^{33}$ . The sun will live  $10^{17}$  more seconds.*



# The “kissing problem”

- ◆ Can 25 unit spheres in four-space be placed so that they are all tangent to the unit sphere centered at origin, and have no more than points of tangency in common?
- ◆ 24 can be so placed, but it is not known if 25 can be so placed.
- ◆ Can be stated with 100 variables.
- ◆  $2^{(2^{100})}$  is *way too big*.



# Theorem Proving in Geometry

- ◆ Wu Wen-Tsen (1986) algebraic methods
- ◆ Since then other algebraic methods have also been used.
- ◆ Good for theorems whose algebraic expressions involve only equalities (not inequalities).
- ◆ Difficult to integrate with graphical software for teaching proofs in geometry.



# Searching for Proofs

- ◆ Problem: avoid generating the entire theory from the axioms, when what you want is the negation of a specific theorem.
- ◆ Solution: the *set of support* strategy
- ◆ Larry Wos (1963)





# Set of Support Strategy

- ◆ Divide the axioms into two lists, *usable* and *set of support* (sos).
- ◆ Put the negation of your theorem in sos
- ◆ Put the axioms in usable.
- ◆ To generate new clauses, use resolution (or a variant of resolution) with one parent from sos and one parent from usable.
- ◆ Afterwards put the sos parent into usable.

# Example

- ◆ Wos (1963)
- ◆ In a group, if  $x*x = e$  for every  $x$ , then the group is commutative ( $z*y = y*z$ ).
- ◆ Today this is trivial (for both humans and computers). In 1962 it was not.







# Using Otter

- ◆ Prepare an input file.
- ◆ Use otter from the command line:
- ◆ `otter < example1.in > example1.out`
- ◆ Otter tells you if it finds a proof.
- ◆ Open `example1.out` to see the proof.

# Equality Reasoning

- ◆ Suppose given a set  $E$  of oriented equations
- ◆ example:  $a*(b+c) = a*b + a*c$
- ◆ *oriented* equations, used left to right only
- ◆  $a, b, c$  matched to complex expressions
- ◆ Keep applying rewrite rules until none apply.  
If this always happens  $E$  is called *terminating*.
- ◆ Is the result unique? If so then  $E$  is called *confluent*.

# Example: Group Theory

Axioms of group theory:

- ◆  $e * x = x$ .
- ◆  $i(x) * x = e$ .
- ◆  $(x * y) * z = x * (y * z)$

Word problem for group theory: *Given an equation, does it follow from these axioms?*

A complete confluent set of rewrite rules would solve this problem. To determine if  $u = v$ , just rewrite  $u$  and  $v$  to normal form and see if the results are identical.



# The Answer for Group Theory

- ◆ The original three equations, plus:
- ◆  $i(x) * (x * y) = y$ .
- ◆  $x * e = x$ .
- ◆  $i(e) = e$ .
- ◆  $i(i(x)) = x$ .
- ◆  $x * i(x) = e$ .
- ◆  $x * (i(x) * y) = y$ .
- ◆  $i(x * y) = i(y) * i(x)$ .



# Knuth-Bendix Algorithm (1967)

- ◆ Input: set  $E$  of equations
- ◆ Output (if algorithm terminates):  
set  $Q$  of rewrite rules which is  
complete and confluent.
- ◆ Algorithm may or may not terminate.
- ◆ Written in Fortran by Bendix.



## A harder example

- ◆ In a ring suppose  $x*x*x = 1$  for all  $x$ .
- ◆ Then  $x*y = y*x$  for all  $x$  and  $y$ .
- ◆ The proof is 52 steps long and takes hours on today's machines.
- ◆ Actually, the hypothesis  $x*x*x = 1$  can be replaced by  $x^n = 1$  for any fixed  $n$ , but it still takes a human being to prove *that*.





# Jan Lukasiewicz (1878-1956) Axiom Systems for Propositional Logic

- ◆  $i(i(x,y),i(i(y,z),i(x,z)))$  (L1)  $i = \text{implies}$
- ◆  $i(i(n(x),x),x)$  (L2)  $n = \text{not}$
- ◆  $i(x,i(n(x),y))$  (L3)
- ◆ To deduce a new formula from  $A$  and  $i(B,C)$ , make substitutions so that  $A' = B'$ , and you can deduce  $C'$ .



# Questions

- ◆ Can you deduce  $i(x,x)$ ?
- ◆ Can we make Otter do proofs like this?
- ◆ Yes, here's how:
- ◆  $P(x)$  means “ $x$  is provable”.
- ◆  $\neg P(x) \mid \neg P(y) \mid P(i(x,y))$ .
- ◆  $P(i(i(x,y),i(i(y,z),i(x,z))))$ .
- ◆  $P(i(i(n(x),x),x))$ .
- ◆  $P(i(x,i(n(x),y)))$ .
- ◆  $\neg P(i(x,x))$ .



# Large Search Space

- ◆ A proof has *level*  $n$  if all its formulas have at most  $n$  symbols.
- ◆ Levels grow like 1, 1, 3, 7, 11, 17, 34, 93, 206, 914, 2806, 41003, 281005,
- ◆ Exhaustive search is impossible
- ◆ *Strategy* is required to find, say, an interesting 50-step proof of level 50.



# Strategies

- ◆ MaxWeight (throw out long formulas)
- ◆ Special inference rules (hyperresolution, UR-resolution, etc.)
- ◆ Hints (some formula forms to keep)
- ◆ Resonance (another way of giving hints)
- ◆ rewriting to “junk” (example,  $n(n(x)) = \text{junk}$ )
- ◆ These strategies have been effective in solving old open problems in logic.



# Automated Deduction as an Art

- ◆ Many strategies have been developed in the last 40 years.
- ◆ Using them together is an art.
- ◆ Otter has many settings and parameters which can be used to control and define search strategies and inference rules.





# Proofs Involving Computations

- ◆ Symbolic computation software such as *Mathematica* and *Maple* is logically incorrect.
- ◆ Example: Set  $a=0$ . Divide by  $a$ . You get  $1 = 0$  since  $a/a = 1$  and  $0/a = 0$ .
- ◆ This prevents just calling *Mathematica* and putting the results into a proof.






# MathXpert and *Weierstrass*

- ◆ MathXpert is symbolic computation software written to be logically correct and produce step-by-step solutions.
- ◆ It is therefore also usable in theorem-proving.
- ◆ I used it in my theorem-prover *Weierstrass* 1998-1990 to advance the state of the art in automated deduction of proofs requiring computation.



# Proofs found with Weierstrass

- ◆ Epsilon-delta proofs of the continuity of specific functions such as powers of  $x$ , square root, log, sine and cosine, etc.
- ◆ Before this, the best that could be done was the continuity of a linear function.
- ◆ Irrationality of  $e$ .



# Proofs by Mathematical Induction

- ◆ Best work has been done by the Boyer-Moore theorem prover, ACL2 (formerly Nqthm).
- ◆ Hard part is to *find* the suitable instance of induction.

# Proofs Involving Simple Sets and Functions, and simple use of numbers.

- ◆ This is my current research, supported by NSF at San Jose State University.
- ◆ Simple example:  $A \times B$  can be put into one-one correspondence with  $B \times A$ .
- ◆ Another example: Lagrange's theorem in group theory: if  $H$  is a subgroup of  $G$ , then the order of  $H$  divides the order of  $G$ .
- ◆ Research involves adding new algorithms to Otter and then using them.



# Reflective theorem-proving

- ◆ It is an art to use Otter.
- ◆ Dozens of parameters to set to control it.
- ◆ Difficult for beginners.
- ◆ Often several runs required.
- ◆ Idea: Let the program adjust its own parameters while it is running.
- ◆ Express rules for this in logic and allow the program to reason about its progress.





# Examples of Reflection

- ◆ This same formula has occurred several times. Let's define it as a new concept and use formulas that contain it to drive new deductions.
- ◆ I've proved one of the lemmas, but not yet the theorem. Let's watch for formulas that look similar to the steps of the lemma, and use them quickly to drive new deductions.





# Spring 2002 Master's Theses at SJSU

- ◆ Nadia Ghamrawi: *Scheme for Automatically Transforming Proofs to Cut-Free Form*
- ◆ Colin Southwood : *Application of Formal Methods to the Analysis of Cryptographic Security Protocols*



# Fall 2002 Master's Theses at SJSU

- ◆ Wei-yi Lin: *Automated Deduction in Circuit Design using Otter*
- ◆ Tony Huang: *Automated Deduction in Ring Theory*
- ◆ Howard Shih: *Automated Deduction in Molecular Modeling using Otter.*



# Wild Speculations

- ◆ Computers may in the future routinely find proofs of new theorems.
- ◆ Computers may in the future identify interesting new conjectures.
- ◆ At some point the combination of these activities may result in original mathematics.